ISTI@TREC Microblog track 2011: exploring the use of hashtag segmentation and text quality ranking

Giacomo Berardi, Andrea Esuli, Diego Marcheggiani, Fabrizio Sebastiani
Istituto di Scienza e Tecnologie dell'Informazione
Consiglio Nazionale delle Ricerche
56124 Pisa, Italy
firstname.lastname@isti.cnr.it

Abstract

In the first year of the TREC Micro Blog track, our participation has focused on building from scratch an IR system based on the Whoosh IR library. Though the design of our system (CipCipPy) is pretty standard it includes three ad-hoc solutions for the track: (i) a dedicated indexing function for hashtags that automatically recognizes the distinct words composing an hashtag, (ii) expansion of tweets based on the title of any referred Web page, and (iii) a tweet ranking function that ranks tweets in results by their content quality, which is compared against a reference corpus of Reuters news. In this preliminary paper we describe all the components of our system, and the efficacy scored by our runs. The CipCipPy system is available under a GPL license.

1 Introduction

Microblogging is a form of personal content sharing that derives from blogging, and it has a focus on concision, space and time locality, and social network-style interactions [5]. While conventional blogging has been for many year the subject of a TREC track [7], this is the first year that microblogging is the subject of a TREC track. Twitter[1] is currently the dominant platform for microblogging, and it has been selected as the source of data for the microblog track.

Twitter enforces a strong policy on short messages, allowing only a maximum length of 140 characters[2]. In addition to simple text, a tweet can contain three types of references:

- URLs to external content. External links are typically used to extend a tweet with other media, e.g., an image shot by the user with her mobile phone, or to point to a web content the tweet comments upon.

[1] http://twitter.com

[2] Twitter text length limitation derives from the original possibility of sending/receiving tweets on cell phones via SMS, which have a maximum length of 160 characters, with 20 characters reserved by Twitter for the id of the author of the tweet.

- Mention of names of other Twitter users, e.g., "*@aesuli*". This is a way to credit the mentioned user for the content of the tweet or a way to establish an exchange of public messages other user may be interested to read and join the discussion.

- *hashtags*: a hashtag is defined by any string prefixed with a #, e.g., "*#whyalwaysme*", "*#iwould*". The string can be a single word, an acronym, or multiple words joined together, and usually identifies the subject topic of the tweet (e.g., "*#SPIRE2011*") or expresses a comment about it (e.g., "*#epicwin*").

In our participation to the microblog track we have focused on using the information contained in linked content and in hashtags in order to improve the quality of a baseline search system that performs a simple search on tweet content using the query text.

We have also explored the use of a text quality ranking measure to filter out "slang" tweets and promote high quality tweets, under the hypothesis that a well written tweet probably contains more relevant information for the user issuing the query.

We have left to future investigation, and to eventual future editions of the track, other equally relevant aspects, e.g., analysing the distribution of tweets along time and geographic locations[3], observing the propagation of tweets in the social network determined by mentions, retweets, and the "follow" relation[4].

The problem of retrieval and ranking in microblogging has been already investigated (see, e.g., [3] [8] [11]), but the 2011 TREC Microblog Track is the first large scale initiative that has focused on microblogging.

2 Downloading the collection

We have downloaded the Twitter corpus using the downloader provided by the organizers. Table 1 reports some statistics on the full corpus and on the various subsets we have defined in order to build our indexes, which are described in Section 3.

The download process took much more time than predicted to complete, notwithstanding that our organization has a fast connection and also that it is closely connected to the Italian internet backbone. If the downloader was able to keep the peak download rate during all the download process, it would have required in theory just five days to complete. In practice, due to various issues, such as crashes in the download process due to lack of memory, hardware failures, variation of the network speed, timeouts connection with Twitter, it required about one month to complete the download.

We have indexed only the tweets in English, filtering out non-English tweets. We have used a language recognition system that we have implemented along the lines of [2], in order to recognize English tweets.

The English set, the only part we have actually indexed, resulted to be roughly a quarter of the entire corpus. We have then indentified two subsets of the English set: a *Hashtag* set

[3]Though only less than 10% of tweets contain location information, according to [10].

[4]We have downloaded the corpus using the HTML crawler, that does not allow us to gather most of this information. For the next year, we plan to redownload the corpus using the Twitter API, that exposes also this information.

	total	effective	retweets	null	hashtags	users
Entire corpus	16.141.812	13.812.346	1.104.780	1.224.686	655.850	5.356.842
English set	4.510.329	4.068.158	442.171		288.753	2.021.759
Hashtags subset	791.464	640.870	150.594		288.753	514.401
Link subset	676.957	674.471	2.486		14.399	400.631

Table 1: Some statistics from the corpus and the subsets we have selected for indexing, the total number of tweets is divided in effective tweets (http status 200), retweets (http status 302) and null tweets (http status 301/403/404). The hashtags column indicates the number of unique hashtags.

that contains only English tweets that have hashtags in them, and a *Link* set that contains only English tweets with URLs in them. The two subsets show a similar ratio of tweets with at least one hashtag/URL with respect to tweets without hashtag/URL (1:5.6 for hashtags, 1:6.6 for URLs).

3 The retrieval system

Our system, named CipCipPy[5], is an indexing and retrieval system based on the Whoosh[6] IR library, written in Python. The source code of CipCipPy is available for download at `http://tag.isti.cnr.it/cipcippy/`.

We have built an index for each of the three set of tweets we have identified in the corpus. We have not indexed retweets, since the guidelines and discussions in the mailing list of the track stated that retweets would be considered not relevant by default.

The index of the English set indexes the text of each tweet as is, without any special processing of the text.

The index of the Hashtag set indexes only the hashtags contained in tweets. Each hashtag is indexed in two ways: (i) the hashtag text "as is", e.g., *#epicfail*, and (ii) the distinct words resulting from hashtag text segmentation (detailed in Section 3.1), e.g., *(epic, fail)*. We choose to index both versions of the hashtags, i.e., unsegmented and segmented, because it is not possible to tell a priori which of the two versions gives the largest contribute of information. For example, the word "skype" extracted from the hashtag #skypeisnotworkingagain, is probably more useful for retrieval than the original hashtag, because the word "skype" clearly identifies the topic of the tweet and has a relatively high statistical relevance in the corpus while the hashtag #skypeisnotworkingagain is a hapax. On the other side, a hashtag like #yeswecan is much more important for retrieval than each of the distinct words composing it.

The index derived from the Link set indexes only the titles (i.e., the text included in the

[5] "Cip Cip" is the Italian word for the sound of birds, while "Py" identifies the Python programming language.

[6] `https://bitbucket.org/mchaput/whoosh/wiki/Home`

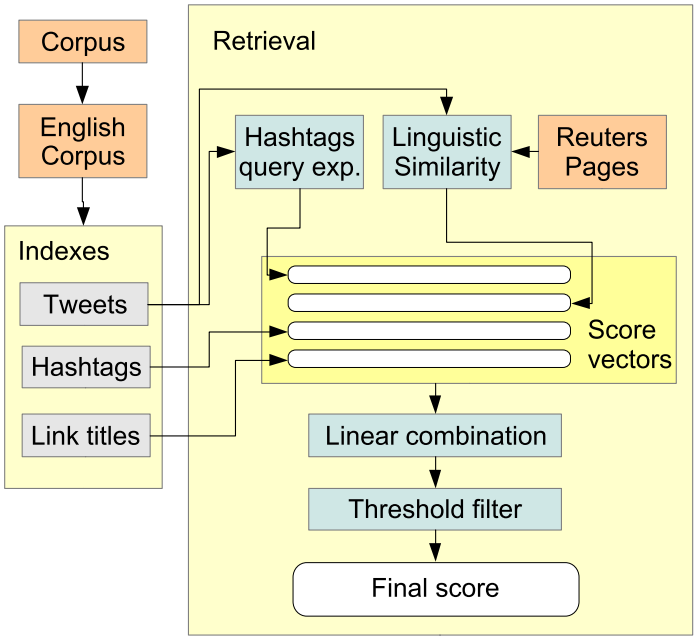

Figure 1: Modules and information flow of CipCipPy

`<title>` tag of the html page) of web pages referred by URLs in tweets. If a tweet contains more than one URL, the titles of the referred pages indexed as consecutive sentences of the same piece of text.

3.1 #splittingwordsinhashtags

A common practice in tweets is to identify their subject topic by means of a hashtag, e.g., #Mancherster, #LiesPeopleAlwaysTell, #toobad, #ff, #skypeisnotworkingagain,. Given that a hashtag cannot contain white spaces, people usually concatenate more words together, to form a short phrase.

Recognizing the distinct words composing a hashtag is not a simple task to be automatized. Some users identify the distinct words using a CamelCase style, i.e., capitalizing the first letter of each word, other leave the words all in lowercase or uppercase. Other use underscores "_" to separate words, but it is not a common case because is wastes characters. Usually words are just juxtaposed without any evident or coherent use of separation signs and actually there are no common rules one can rely on to segment hashtags.

We treated our hashtag splitting problem as a compound word segmentation problem. Choosing a sequence of words of sense given a word compound is difficult even for a human, for example the hashtag compound *#airportend* can be split as *air portend*, or as *airport end*, and also as *air port end*. The method we used to solve word compound segmentation is based on the Viterbi Algorithm [4]. As the model of words distribution, used by the Viterbi algorithm to calculate the most probable sequence, we have taken words frequencies from

an English corpus[7].

Given the words distribution model and a hashtag, the hashtag segmentation module converts the hashtag to a vector of words composing them (eventually returning just a single word equal to the original hashtag).

3.2 Ranking by text quality

Quality of microblog content can be rather poor. Twitter content refers often to personal topics, so the language style is grammatically and syntactically incorrect, conversational, full of "vulgar" expressions. Only a small portion of tweets comes from authoritative sources [1]. Following the hypothesis that a well written tweet has more probability to be relevant for a query, we designed a ranking function to filter out unreliable tweets and increase the importance of well formed tweets.

We have designed a simple method to assign a score of linguistic quality to a text, following an approach that is typical of authorship attribution literature [6]. We represent any text document, i.e., a tweet, with a vector of linguistic features:

- Ratio between the unique POS[8] (parts of speech) and the total number of possible POS (i.e., 36, from the NLTK), averaged on sentences.
- Ratio between the unique words and total number of words, averaged on sentences.
- Average length of sentences, in terms of words.
- Average length of words.

With this representation we can compare two texts on these linguistic features, so the more similar a tweet vector is to a well written text vector, the more the tweet quality is likely to be high.

In our retrieval system, for each query, we retrieve the first r ($r = 10$) results from the Reuters website[9], using Google to search on that website. We consider that set of result to be an example of well written content on the topic. A single linguistic vector from the union of these result pages is created, and then compared with each vector of the tweets retrieved with the same query. We have used the classic Cosine similarity as the similarity function, thus the higher is the score, the higher we consider the linguistic quality of a tweet.

3.3 Query processing

For the official runs we have defined a weight function (used to compute the retrieval score of the tweets) that is a variant of the BM25 function [9]. The difference between our function and the BM25 function is in the computation of term weight. In the BM25 approach, frequency of the term in a document is used, while in our approach we use the inverse of

[7] `http://norvig.com/big.txt`

[8] We have used the Natural Language Toolkit (`http://www.nltk.org/`) to perform POS-tagging.

[9] `http://www.reuters.com/`

the frequency of the term in a tweet. Given a query q composed by terms $q_1...q_n$, the score assigned to document d is:

$$score(d,q) = \sum_{i=1}^{n} IDF(q_i) \cdot \frac{(1/TF(q_i,d)) \cdot (k_1+1)}{(1/TF(q_i,d)) + k_1 \cdot (1 - b + b \cdot \frac{|d|}{avgdl})} \quad (1)$$

where $avgdl$ is the average length of the documents in the collection, k_1 and b are parameters set to the default values of the whoosh library ($k_1 = 1.2$ and $b = 0.75$). The IDF function returns the inverted document frequency of a term:

$$IDF(q_i) = log \frac{(N - |\{d : q_i \in d\}|) + 0.5}{|\{d : q_i \in d\}| + 0.5} \quad (2)$$

where N is the total number of documents in the collection. The intuition is that in a short text, composed of few unique words, the repetition of the same word is an indicator of poor quality of the text.

We have used hashtags in order to implement a simple result re-ranking method. Given the original query q, we first get the top k results ($k = 30$) for the English set index, and we extract all the hashtags we find in these results. Then we use our hashtag segmentation method (see Section 3.1) to build a vector h of relevant hashtag-related words. Each word in the vector h is weighted by its frequency in the k results described above. The vector is then normalized by the L^2 norm. The h vector is thus designed to identify a weighted set of keywords that describe the topic of the query.

We have also adopted a query expansion method based on hashtags. An expanded query q is formulated as the union, based on the ANDMAYBE[10] Whoosh operator, of the original query q with the vector h (in which each term is considered to be in disjunction with the others). In this way the score a tweet receives for matching the original query q is increased with the score it receives for matching with any of the words in the weighted vector h. The q' query does not retrieve more tweets than the q query, but it is designed to push the rank of tweets that have matches with the h vector.

We retrieve n results (n=1000) from each of the three indexes using the q query. At this point, we have three score vectors, of size n.

In the successive phase we compute the cosine similarity between the linguistic vectors of tweets, retrieved from the first index, and the Reuters pages, retrieved using the query q, as described in Section 3.2.

Each of the four score vectors is then normalized using L^2 norm. The four score vectors are linearly combined, with parameters w_{tweet}, $w_{hashtag}$, w_{title} and $w_{quality}$. After removing duplicates (i.e. identical tweets posted after the original), we filter out tweets with score values below a threshold parameter α. We have empirically set the α value to 0.03, that results in filtering out a relatively small part of low-scored tweets (from 1 to 10% for each query). The remaing tweets, ordered by time, are the answer to the query.

[10] `http://packages.python.org/Whoosh/api/query.html#whoosh.query.AndMaybe`

4 Official runs

We have been able to submit only two official runs, *runNeMIS* and *runNeMISext*.

The runNeMIS run only uses "internal" information, i.e., the weights w_{title} and $w_{quality}$ are set to zero, while $w_{tweet} = w_{hashtag} = 0.5$. In this way only the information contained in tweets is used to perform the search.

The runNeMISext uses also the external information, i.e., $w_{tweet} = w_{hashtag} = w_{title} = w_{quality} = 0.25$.

Tables 2 and 3 resumes the precision@30 values scored by the two run, compared with the baseline run, the best and median performance values made available to us by the organizers. Results indicates that our system performs worse than the median reference, which is clearly not a satisfactory result. It is interesting to note that, as expected, the run with external information performs better than the run that does not use such information.

	runNeMIS	runNeMISext	Baseline	Best Average	Median Average
P30	0.139	0.171	0.099	0.612	0.259

Table 2: AllRel evaluation

	runNeMIS	runNeMISext	Baseline	Best Average	Median Average
P30	0.049	0.062	0.106	0.265	0.069

Table 3: HighRel evaluation

5 Post-conference runs

After the TREC conference we have collected feedback on the system and modified it in order to improve it, producing additional runs. First, we have better explored the space of parameters (i.e., k, n, r, w_*, α).

One relevant modification to the system is the use of a simple IDF weighting function instead of our custom BM25 weighting. After a manual inspection of our official runs, we hypothesized that the BM25 measure could be not well suited for the task, given the short and relatively compact distribution of the text lengths. We then adopted for the additional runs the IDF measure for weighting.

In addition to the already described scores, we have added a date score vector, in which the values are proportional to the time difference between the query date and the date of the tweets. This result vector for dates is normalized, similarly to the others, by the L^2 norm. The final tweet score is the linear combination of all the previous scores, multiplied by the relative parameters, plus the date score multiplied by w_{date}.

Considering the low recall of the official runs, the queries have been reformulated to increase the recall of the system. Instead of a conjunction of all the terms, the q query is

defined as the disjunction of the conjunction of every pair of terms of the official query, plus each single term. We have then simplified the query expansion method based on hashtags and the ANDMAYBE operator, combining instead the results retrieved with the new q query with the results retrieved using the the hashtag terms in the h vector as a disjoint query.

	runNeMISNew1	runNeMISNew2	runNeMIS	runNeMISext
MAP	0.252	0.249	0.096	0.119
P5	0.506	0.489	0.346	0.326
P30	0.355	0.352	0.139	0.171

Table 4: Result evaluations for three new runs and the old runs (AllRel), with the measures of Mean Average Precision (MAP), precision@5 (P5) and precision@30 (P30).

In the Table 4 a selection of two runs with a setting of parameters that scored good results are shown. We'd like to stress that these new runs are somewhat optimized on test data, so we don't look for any comparison with official runs from other participants, but we consider them just as an indicator of the potential of the system when it is possible to fine tune it. We have found that a slightly higher value of $\alpha = 0.04$ produces a relevant improvement in precision, and all the new runs have this filtering threshold. In runNeMISNew1 the other parameters are $k = 5$, $w_{tweet} = 0.8$, $w_{title} = w_{hashtag} = w_{date} = 0.07$ and $w_{quality} = 0$, this is one of the best run in terms of P30. The runNeMISNew2 run is very similar to the runNeMISNew1 run in terms of efficacy; the only non zero weight is $w_{tweet} = 1$, and $k = 0$, so this run is equivalent to a simple text retrieval using only the IDF weighting, and the new q query formulation.

In general, we have found that rather different configurations of parameter values produced similarly good results, but we have not yet found a dominant component in the configurations.

6 Conclusion and future work

Creating a system for a TREC track is often a difficult task, and this year made no difference. We regard as a good result of our participation the successful development, from scratch, of a complete retrieval system, with some interesting modules designed ad-hoc for the task.

The development effort unfortunately limited us from performing a complete and rigorous exploration of our hypotheses and proposals, and lead to poor results. Many of the parameters of the system have been hardcoded to rather arbitrary values, determined by a few manual inspections of preliminary results on the sample topics released before the official topics.

A first exploration of the behaviour of the system in relation to the variation of the parameters (e.g., k, n, r, w_*, α), shown that the system has a good potential for improvement.

A simplification of the weighting function, and a reformulation of the queries in order to increase recall produced a relevant improvement.

In the future we would like to expand the system to consider additional sources of information, such as the distribution in space of tweets, their propagation along the Twitter social network, and also to include the best ideas from other participants.

References

[1] P. Analytics. Twitter study, 2009. Retrieved 03 31, 2010, from http://www.scribd.com/doc/18548460/Pear-Analytics-Twitter-Study-August-2009.

[2] W. B. Cavnar and J. M. Trenkle. N-Gram-based text categorization. In *In Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, 1994.

[3] M. Efron. Information search and retrieval in microblogs. *Journal of the American Society for Information Science and Technology*, 62(6):996–1008, 2011.

[4] G. Forney Jr. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[5] A. Java, X. Song, T. Finin, and B. Tseng. Why we twitter: understanding microblogging usage and communities. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, WebKDD/SNA-KDD '07, pages 56–65, New York, NY, USA, 2007. ACM.

[6] P. Juola. Authorship attribution. *Foundations and Trends in Information Retrieval*, 1(3):233–334, 2006.

[7] C. Macdonald, R. L. Santos, I. Ounis, and I. Soboroff. Blog track research at trec. *SIGIR Forum*, 44:58–75, August 2010.

[8] R. Nagmoti, A. Teredesai, and M. De Cock. Ranking approaches for microblog search. In *Proceedings of the 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 01*, WI-IAT '10, pages 153–157, Washington, DC, USA, 2010. IEEE Computer Society.

[9] S. Robertson and H. Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3:333–389, 2009.

[10] S. Vieweg, A. L. Hughes, K. Starbird, and L. Palen. Microblogging during two natural hazards events: what twitter may contribute to situational awareness. In *Proceedings of the 28th international conference on Human factors in computing systems*, CHI '10, pages 1079–1088, New York, NY, USA, 2010. ACM.

[11] J. Weng, E.-P. Lim, J. Jiang, and Q. He. Twitterrank: finding topic-sensitive influential twitterers. In *Proceedings of the third ACM international conference on Web search and data mining*, WSDM '10, pages 261–270, New York, NY, USA, 2010. ACM.